Project 1 – DPLL/SAT

CIS700 — Fall 2023

Kris Micinski

This project has a fairly short specification.

Your job is to build a SAT solver.
- Your SAT solver should be invoked on the command line. You may write in any language you want: Python, C++, Racket, Haskell, etc…
- Your SAT solver should take a single argument, which is a file in DIMACS CNF input format. The format is very simple—it should be possible to parse it by reading each line and using a string split operation
- Your program should write an output to the console of either:
  - (a) the string “UNSAT” or (b) a satisfying assignment to the formula.

DIMACS CNF Input Format

Read the following short guide:

https://jix.github.io/varisat/manual/0.2.0/formats/dimacs.html

Example (from that site): encode (x ∨ y ∨ ¬z) ∧ (¬y ∨ z) as

```
p cnf 3 2
1 2 -3 0
-2 3 0
```

First line specifies number of propositions (sequentially numbered), followed by number of clauses

Each subsequent line gives a clause, consisting of whitespace-separated literals (either l, positive l or -l, not l), ending in 0 (end of line sentinel)

Projects are due on the 28th, anywhere on earth (so, 7am on the 29th in syracuse).

You may work in groups of up to three. If you are relatively more advanced, please try to work in a group of students which is less confident.

If you are less interested in dedicating time to the class, it is acceptable to do the baseline of simply enumerating the SAT instances and checking all of them.

If you are feeling you seriously want to learn this material implement either (a) DPLL or (c) CDCL

Submissions and Grading

Please email me submissions, kkmicins@syr.edu, CCing all of your group mates.

I will test your submissions with a variety of DIMACS inputs, e.g., the ones from this page. https://people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html

Please tell me how to invoke and run your program when you email me. Ensure that it can run on either a Mac or Linux machine (I have both of these)—give me sources and instructions to build your project.

Grading / Rubric

- I will give you a 95% if your project is a good-faith attempt that is a working solution for small CNF instances. If you get occasional hiccups, I am fine with that. If you get a substantial number of instances incorrect (i.e., wrong model/ unsat) I will consider taking off some points. Ensure that your implementation is correct—I will grade whether or not you have tests for your project. I expect **some** testing infrastructure, even if it is relatively basic—I am not expecting production quality, but a convincing argument that it was constructed with an eye to correctness
- I will give you +5% if you implement more than enumerate-and-check, i.e., if you implement DPLL or something similar
- I will give you +5% *bonus* (so, 105%) if you implement CDCL (which we will discuss next week) as well. And I will be very impressed.